Deep-Learning in Content-Based Recommender Systems

Nicole Petrozzo

Advised by Professor Douglas Blank

Extended Abstract

1 Introduction

Nice and concise definition of recommender systems

Recommender systems predict a user's opinion on various items, outputting items with the highest ratings or relevance as personalized recommendations for that user. These systems are an integral part of many web-based businesses, like Amazon and Netflix, because they are able to produce recommendations from a large catalog of products that would be too tedious for a user to survey on their own.

Most recommender systems operate using one of two main methods: collaborative filtering, or content-based filtering. Recommender systems that use collaborative filtering produce recommendations purely using user data: mainly, the items a user has used, and the user's opinion on each item. These types of systems output recommendations based on comparisons of user data; if two users have similar taste, then they are more likely to like the same items. Good explanation of limitations A major drawback of collaborative filtering is that the quality of the recommendations produced is dependent on the amount of user data available. A new system with very few users will encounter a "cold start" problem, where the system does not have enough data to produce reliable recommendations [1]. Furthermore, this type of system does not take item attributes into account when creating recommendations.

Clear description

Recommender systems that use content-based filtering avoid this problem by using item meta-data to produce recommendations, rather than solely relying on user data. When a new user first registers, they can be prompted with simple questions used to gauge their preferences, which can then be matched with item descriptions to immediately give relevant recommendations. However, content-based recommender systems are arguably more limited in scope, as they will only recommend items that are similar to ones that the user has already seen.

The main way in which content-based recommender systems may differ from each other is how they construct user profiles. User profiles are models of each user that inform the system on their preferences, usually in the form of a set of vectors. These preferences can then be matched to item descriptions to give relevant recommendations to the user.

A common, effective approach is to use a naive Bayesian classifier to learn a user's preferences [2]. Another approach, with similar performance, is to use decision trees, particularly the C4.5 algorithm [2]. These examples will be discussed more in-depth in Chapter 3: Background.

Clear thesis topic!

This thesis proposes a content-based recommender system that uses deep neural networks to produce recommendations. The system would be trained using existing users, their ratings on existing items, and the descriptions for each item. After being trained on this data, the system will ideally be able to build profiles for users and items that it has never been trained on, and give reliable recommendations based on these profiles. The expected advantages of this system are:

Give a brief intro or better example of the "cold start" problem!

1. It does not solely rely on user data, thereby circumventing the "cold start" problem.

2. By using neural networks with several hidden layers, it will be able to detect attributes and similarities in items beyond what the existing meta-data would suggest, unlike systems that use naive Bayes to predict user-modeling. This would be particularly useful for systems that recommend non-text items such as audio and video, because it is more difficult to extract attributes that describe these items.

Maybe briefly explain why and how? If you feel like to? After reading, I would love to get a little more info on this

3. Because it uses machine learning, the system would constantly be improving over time as more data is added.

4. Although the system does not solely rely on user data to produce recommendations, the neural networks may pick up on attributes about different users during training. In this way, the system would be something of a hybrid between collaborative-filtering systems and content-based systems, overall producing more informed and relevant results.

2 Methodology

The proposed system for this thesis would be built in Python 3.4.6, using the **conx** package to build and train neural networks, and the **pandas** package to parse data. The data that will be used to train and evaluate this system is the MovieLens 10M Dataset [3]. This dataset was released January 2009, and contains real, anonymized data: 10 million ratings and 10,000 tags for 10,000 movies, created by 72,000 users. All users for the data had rated at least 20 movies, but were otherwise randomly selected. Each movie in the dataset has been labeled by genre, as well as with user-generated tags. The system proposed in the thesis will, given an input of a known user and a movie the user has not yet rated, output a predicted rating that user would give for that movie. All code will be written and run within a Jupyter notebook.

The minimum goal of this thesis is to produce such a system as described above, and evaluate its outputs using NDCG, a measure of evaluating quality based on relevance. The ideal goal of this thesis is to also build two other content-based recommender systems, and perform a comparative analysis of all three systems. These two other systems would be simple, using more traditional methods of content-based filtering: one would use a Naive Bayes classifier, and the other would use the C4.5 decision tree algorithm. If time permits, these algorithms would be implemented in Python 3.6.4 using the **scikit-learn** package.

3 Background

In this section, we will further discuss recommender systems and deep learning, as well as review related works done in these fields.

3.1 Approaches to User-Modeling

An important component of content-based recommender systems is user-modeling. User models describe a set of attributes or values pertaining to a particular user's preferences. Using these models, the system can then offer personalized recommendations by finding items whose features are relevant to that user's preferences. Modern systems do not require manual user-modeling, but instead use a variety of classification algorithms to predict a model for a given user.

3.1.1 Naive Bayes Classifiers

A common approach to user-modeling implements a naive Bayes classifier. A naive Bayes classifier is a technique of predicting the classification of an unknown data-set. To effectively use naive Bayes, an engine is first trained on a training set consisting of a list of instances and their attributes. Based on the instances from training set, the system can then compute the most probable classification for a new instance.

The formula for the naive Bayes classifier, given attribute values $< a_1, a_2, ..., a_n >$, is as follows:

$$v_{NB} = argmax P(v_j) \prod_i P(a_i|v_j)$$

where V is a finite set of attribute values, and $v_j \epsilon V$.[4]

In the case of recommender systems, naive Bayes can be applied to user-modeling by using a training set of items to be recommended as instances, and descriptions or meta-data of those items as their attributes. The naive Bayes classifier can then be used to predict a user's preferences on various attributes. For example, if Alice has a history of liking movies of the horror genre, she is more likely to like other horror movies in general.

Good example to clarify

The disadvantage of this approach is that it assumes that all features of an object are independent. In many cases, especially for recommender systems, this is simply not true. For example, perhaps Alice usually likes horror movies, but she only likes horror movies that came out before 1990. This approach wrongly assumes that Alice likes all horror movies, regardless of the year of release.

You must mean independent probablistic-wise / statistically. Maybe say a word or two to clarify `independent` for people without probability background?

3.1.2 Decision Trees

Another popular method of user-modeling in content-based recommender systems uses decision trees. In this method, a tree is built for each user, with the user as the root. From the root, different arcs are added, representing different attributes. The arcs connect to sub-nodes, which represent different attribute categories. Figure 1 shows an example decision tree user model for Alice.

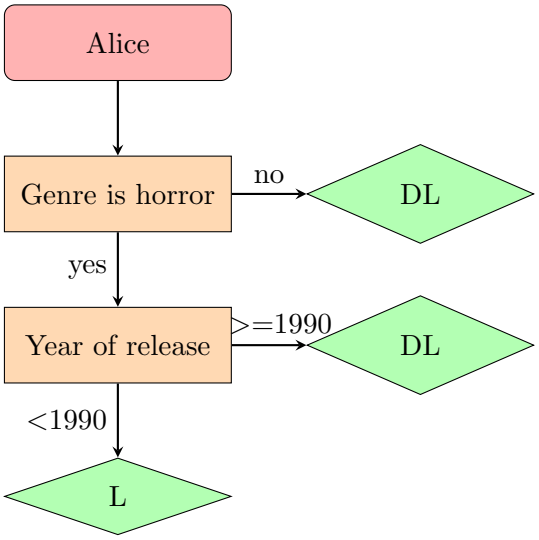

FIGURE 1: A simple user model based on a decision tree.

In Figure 1, we can see each subnode of Alice represents some attribute category, while the arcs represent the attribute values themselves. The decision nodes are "L" and "DL" for "like" and "dislike", respectively. According to this model, Alice only likes movies that fit into the horror genre and were released before 1990.

One of the most popular algorithms for constructing decision trees is the C4.5 algorithm [5]. The system first uses a training data-set: a list of already classified values. In building the tree, the normalized information gain is calculated for splitting from each attribute. The algorithm then creates subnodes based on the highest normalized information gain for each attribute, and then recurs on the sublists created by each split.

Like the user-models built using naive Bayes, C4.5 can also make predictions for unknown items based on a user model.

The drawback of using C4.5 for user-modeling is that there is a large computational overhead. A tree must be constructed for every single user. Then, to produce a recommendation requires a traversal of the tree from root to leaf for each item. Therefore, this method is not suitable for systems dealing with large amounts of data.

Gershman and Meisels propose a new decision tree algorithm for recommendations that contains a pre-computed recommendation list at the leaf-nodes, therefore improving the

computation time. [6]. However, this algorithm will still run into an over-fitting problem when used with small datasets.

3.2 Deep Learning

Deep learning is a relatively new field in machine learning that uses "deep neural networks": neural networks with several hidden layers between the input and output layers. Deep neural networks are very useful for modeling complex, non-linear relationships where the relationship has many uncertainties. In this section, we provide an overview of the basics of deep learning, and discuss current applications of deep learning in recommender systems.

3.2.1 Overview

Neural networks are algorithms designed to recognize patterns. Deep neural networks are a sub-category of neural networks that use multiple hidden layers. Figure 2 demonstrates the difference between a simple neural network (one with only one hidden layer), and a deep neural network.

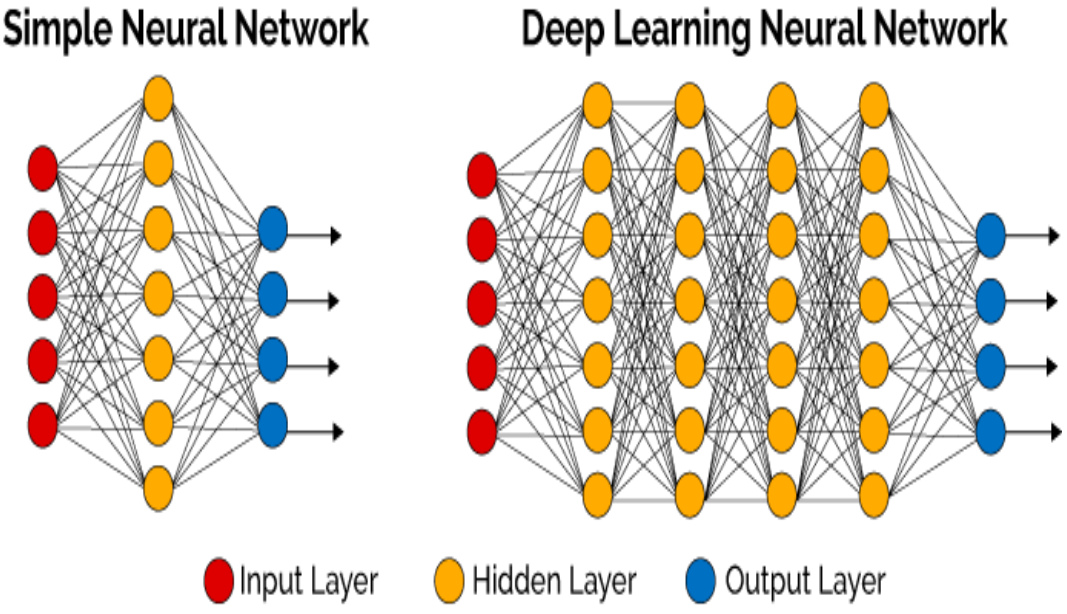

FIGURE 2: A diagram showing the difference between simple neural networks and deep neural networks, from Jagreet Kaur Gill [7].

Nice clarification with the graph!

In particular, deep neural networks are powerful because they can perform automatic feature extraction. This makes them useful in applications with large amounts of unstructured data.

3.2.2 Applications in Recommender Systems

The ability of deep neural networks to recognize patterns in large amounts of unlabeled, unstructured data makes them useful for building recommender systems.

One such application is described by van den Oord, who uses neural networks to generate music recommendations [8]. What is striking about this project is that they do not use any human-readable data to describe the songs, but instead they use a bag-of-words representation of the audio signals themselves. The trained networks are able to give sensible recommendations, and even outperform some traditional methods of music recommendation. However, this method processes the user ratings as independent from the audio content information. It is likely that processing the relationship between user-item rating and item content would improve the recommendations even more.

In another example, Bansal uses deep recurrent neural networks to recommend scientific papers [9]. More specifically, they use neural networks to map an item's text content to a vector of latent factors. Bansal's approach combines the vector representations of each item as outputted by the neural networks with user embeddings based on the user's ratings. They found that the predictions produced in cold-start scenarios improved on even state-of-the-art systems.

I like how you summarize the background, indeed lead readers through definitions, explanations, examples and limitations!

4 Approach

4.1 Current Progress

The progress done on this project so far has mainly consisted of background reading related to recommender systems, especially those that implement deep learning to create recommendations. The field of recommender systems is rich, with many different proposed approaches of optimization and personalization. The motivation for this project is not only academic interest in a new and growing topic, but to propose a unique system that ideally improves on past implementations. We now feel comfortable discussing and analyzing content-based recommender systems, and we feel more knowledgeable on current progress done in the field.

What are you trying to convey in the last sentence?

Based on my understanding, you seem to talk about personal experience and thoughts on the current progress. Maybe summarize this paragraph in a different way or word it more formally

We have also built a simple recommender system using conx. The system does not yet use movie meta-data in any way. However, it can make recommendations based on user ratings. The neural network uses an embedding layer to learn to represent users and movies in such a way that similar users are represented similarly, and similar movies are represented similarly. The quality of these recommendations has yet to be evaluated.

4.2 Next Steps

Very clear next step!

Our next steps are to build a recommender system that uses the genre and tags data available for each movie. One step we have not yet decided on is how to create movie profiles for the neural network. We have already decided that the neural network will take an input a user profile and a movie profile, the latter being based on the tags available. The original idea was to input a list of vectors, where each vector represents a tag. However, any given movie in the dataset can have any number of tags; some may only have one, and some may have several. We are not yet sure how to normalize this data to create a list of vectors.

Bibliography

[1] Francesco Elahi, Rubens Mehdi, and Neil Ricci. A survey of active learning in collaborative filtering recommender systems. *Computer Science Review*, June 2016. URL https://www.researchgate.net/publication/303781992_A_survey_of_active_learning_in_collaborative_filtering_recommender_systems.

[2] Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27:313–331, November 1997. URL http://www.ics.uci.edu/~pazzani/Publications/SW-MLJ.pdf.

[3] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5, December 2015. URL http://dx.doi.org/10.1145/2827872.

[4] Mia K. Stern, Joseph E. Beck, and Beverly Park Woolf. Naive bayes classifiers for user modeling. 1999. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.47.1676&rep=rep1&type=pdf.

[5] Peng Li and Seiji Yamada. A movie recommender system based on inductive learning. December 2004. URL http://136.187.116.132/lab/publication/conference/2004/CIS-2004-li.pdf.

[6] Amir Gershman and Amnon Meisels. A decision tree based recommender system. *IICS*, pages 170–179, June 2010. URL https://subs.emis.de/LNI/Proceedings/Proceedings165/170.pdf.

[7] Jagreet Kaur Gill, 2017. URL https://www.xenonstack.com/blog/data-science/log-analytics-with-deep-learning-and-machine-learning. [Online; accessed February 20, 2018].

[8] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. pages 313–331, 2013. URL http://papers.nips.cc/paper/5004-deep-content-based-music-recommendation.pdf.

[9] Trapit Bansal, David Belanger, and Andrew McCallum. Ask the gru: Multi-task learning for deep text recommendations. 2016. URL https://arxiv.org/pdf/1609.02116.pdf.